Exclusive Perpetual Ring Exploration without Chirality

Lélia Blin[†] Alessia Milani[‡] Maria Potop-Butucaru[‡,⋆] Sébastien Tixeuil[‡]

[†]Université d'Evry, IBISC, France.
[‡]Univ. Pierre & Marie Curie - Paris 6, LIP6-CNRS UMR 7606, France.
[⋆]INRIA REGAL, France.

Abstract

In this paper, we study the exclusive perpetual exploration problem with mobile anonymous and oblivious robots in a discrete space. Our results hold for the most generic settings: robots are asynchronous and are not given any sense of direction, so the left and right sense (*i.e.* chirality) is decided by the adversary that schedules robots for execution, and may change between invocations of a particular robots (as robots are oblivious). We investigate both the minimal and the maximal number of robots that are necessary and sufficient to solve the exclusive perpetual exploration problem. On the minimal side, we prove that three deterministic robots are necessary and sufficient, provided that the size n of the ring is at least 10, and show that no protocol with three robots can exclusively perpetually explore a ring of size less than 10. On the maximal side, we prove that $k = n - 5$ robots are necessary and sufficient to exclusively perpetually explore a ring of size n when n is co-prime with k.

1 Introduction

We consider autonomous robots that are endowed with visibility sensors (but that are otherwise unable to communicate) and motion actuators. Those robots must collaborate to solve a collective task, namely *exclusive perpetual exploration*, despite being limited with respect to input from the environment, asymmetry, memory, etc. In this context, the exclusive perpetual exploration tasks requires every possible location to be visited infinitely often by every robot, with the additional constraint no two robots may be present at the same node concurrently.

Robots operate in *cycles* that comprise *look*, *compute*, and *move* phases. The look phase consists in taking a snapshot of the other robots positions using its visibility sensors. In the compute phase a robot computes a target destination based on the previous observation. The move phase simply consists in moving toward the computed destination using motion actuators.

The robots that we consider here have weak capacities: they are *anonymous* (they execute the same protocol and have no mean to distinguish themselves from the others), *oblivious* (they have no memory that is persistent between two cycles), and have no compass whatsoever (they are unable to agree on a common direction or orientation in the ring).

Related works. While the vast majority of literature on coordinated distributed robots considers that those robots are evolving in a *continuous* two-dimensional Euclidian space and use visual sensors with perfect accuracy that permit to locate other robots with infinite precision, a recent trend was to shift from the classical continuous model to the *discrete* model. In the discrete model,

space is partitioned into a *finite* number of locations. This setting is conveniently represented by a graph, where nodes represent locations that can be sensed, and where edges represent the possibility for a robot to move from one location to the other. Thus, the discrete model restricts both sensing and actuating capabilities of every robot. For each location, a robot is able to sense if the location is empty or if robots are positioned on it (instead of sensing the exact position of a robot). Also, a robot is not able to move from a position to another unless there is explicit indication to do so (*i.e.*, the two locations are connected by an edge in the representing graph). The discrete model permits to simplify many robot protocols by reasoning on finite structures (*i.e.*, graphs) rather than on infinite ones. However, as noted in most related papers [6, 5, 3, 4, 2], this simplicity comes with the cost of extra symmetry possibilities, especially when the authorized paths are also symmetric.

Assuming visibility capabilities, the two main problems that have been studied in the discrete robot model are gathering [6, 5], exploration with stop [3, 4, 2], and exclusive perpetual exploration [1]. For exploration with stop, the fact that robots need to stop after exploring all locations requires robots to "remember" how much of the graph was explored, *i.e.*, be able to distinguish between various stages of the exploration process since robots have no persistent memory. As configurations can be distinguished only by robot positions, the main complexity measure is then the number of robots that are needed to explore a given graph. The vast number of symmetric situations induces a large number of required robots. For tree networks, [4] shows that $\Omega(n)$ robots are necessary for most n-sized tree, and that sublinear robot complexity is possible only if the maximum degree of the tree is 3. In uniform rings, [3] proves that the necessary and sufficient number of robots is $\Theta(\log n)$, although it is required that the number k of robots and the size n of the ring are coprime. Note that both approaches are *deterministic*, *i.e.*, if a robot is presented twice the same situation, its behavior is the same in both cases. In [2], the authors propose to adopt a *probabilistic* approach to lift constraints and to obtain tighter bounds. They show that *four* identical probabilistic robots are necessary and sufficient to solve the exploration problem in any anonymous unoriented ring of size $n > 8$, also removing the coprime constraint between the number of robots and the size of the ring. Most related to our work is the exclusive perpetual exploration for grids and partial grids presented in [1]. While this paper considers perpetual exploration instead of exploration with stop, it introduces the additional constraint that no two robots should ever concurrently be located at the same node or cross the same edge (denoted in the following as the *exclusivity* property). Moreover, differently from the traditional perpetual exploration, the exclusive perpetual exploration, requests that each robot visits infinitely many times each node of the ring. The authors investigate the maximum number of robots that can perpetually explore a partial grid under such conditions and in a synchronous model. Contrary to [6, 5, 3, 4, 2], robots are endowed with sense of direction, *i.e.* they agree on the four basic directions: north, south, east, and west. This technique obviously permits to break all cases of initial symmetry since a global total order can be inferred on nodes.

Our contribution In this paper, we initiate research about exclusive perpetual exploration with mobile anonymous and oblivious robots in the discrete model in ring-shaped networks. Contrary to [1], our robots are not given any sense of direction, so the left and right sense (*i.e.* chirality) is decided by the adversary that schedules robots for execution, and may change between invocations of a particular robots (as robots are oblivious). This very weak assumption preserves all usual problems related to symmetry breaking. We investigate both the minimal and the maximal number of robots that are necessary and sufficient to solve the exclusive perpetual exploration problem. On

the minimal side, we prove that three deterministic robots are necessary and sufficient, provided that the size n of the ring is at least 10, and show that no protocol with three robots can exclusively perpetually explore a ring of size less than 10. On the maximal side, we prove that $n-5$ robots are necessary and sufficient to exclusively perpetually explore a ring of size n when n is coprime with k.

2 Model

We consider a distributed system of mobile robots scattered on a ring of n nodes u_0,u_1,..., $u_{(n-1)}$ such as u_i is connected to both $u_{(i-1)}$ and $u_{(i+1)}$. The ring is assumed to be anonymous *i.e* there is no way to distinguish the nodes or the edges (i.e. there is no available labeling). In addition, the ring is unoriented *i.e* given two neighbors, it is impossible to determine which node is on the right or on the left of the other. On this ring k robots collaborate to explore all the nodes of the ring. The robots are identical *i.e* they cannot be distinguished using their appearance and all of them execute the same protocol. Additionally, the robots are oblivious *i.e* they have no memory of their past actions. We assume the robots do not communicate in a explicit way. However, they have the ability to sense their environment and see the position of the other robots. Robots operate in three phase cycles: Look, Compute and Move. During the Look phase robots take a snapshot of their environment. The collected information (position of the other robots) are used in the compute phase in which robots decide to move or to stay idle. In the last phase (move phase) they may move to one of their adjacent nodes towards the target destination computed in the previous phase.

At some time t, a subset of robots are activated by an abstract entity called *scheduler*. The scheduler can be seen as an external entity which selects some robots for the execution. In the following we assume that the scheduler is fair *i.e* each robot is activated infinitely many times. Two computational models exist: The *ATOM model* [8], in which synchronous cycles are executed in atomic way *i.e* the robots selected by the scheduler at the beginning of a cycle execute synchronously the full cycle, and the *CORDA model* [7] in which the scheduler is allowed to interleave different phases (For instance one robot can perform a look operation while another is moving). The model considered in our case is the CORDA model with the following constraint: the Move operation is instantaneous *i.e* when a robot takes a snapshot of its environment, it sees the other robots on nodes and not on edges. Nevertheless, since the scheduler is allowed to interleave the operations, a robot can move according to an outdated view (during the computation phase, some robots have moved).

In the following we assume that initially every node of the ring contains at most one robot. During the system execution a subset of robots are activated and move to other nodes. A robot that actually moves to an adjacent node when activated by the scheduler is said *activatable*. The position of all the robots at time t is the system configuration at t. During the Look phase, the activated robots take a snapshot of their environment in order to see the position of the other robots.

3 A Protocol with 3 robots

In this section we propose a protocol that achieves a perpetual exploration of a ring of size $n \geq 10$ with 3 robots. As shown in the appendix, three robots are the minimal number of robots that can solve the exclusive perpetual exploration problem. We identify two types of configurations:

legitimate (configurations reachable during the perpetual exploration) and non legitimate (e.g. initial configurations).

When started in a legitimate configuration the protocol always moves the system in a legitimate configuration. When started in a non-legitimate configuration the protocol ensures the convergence towards a legitimate configuration. For the sake of the presentation we divide the protocol into two phases: the first phase is executed whenever the system is in a legitimate configuration while the second phase works when the protocol is in a initial configuration. The protocol divides into two phases following the type of the current configuration. The actions of the protocol are divided following the type of configuration we consider.

In the following a configuration is characterized by the distances (counted in terms of empty nodes) between robots. Let (x, y, z) denote the class of configurations where the distances between the three robots are x, y respectively z. Since robots do not have chirality, in all these configurations they will execute the exactly same actions therefore all these configurations can be treated as one.

3.1 Phase I

We identify the following legitimate configurations $C_0 = (0, 2, z)$, $C_1 = (1, 2, z)$ or $C_2 = (0, 3, z)$ with $z \notin \{0, 1, 2, 3\}$. These configurations will be referred in the following as *two-gap*, *one-two-gap* and *three-gap* configurations respectively. The first phase of the algorithm makes the system cycle between these legitimate configurations. A two-gap configuration moves to a one-two-gap configuration (via Rule 1), a one-two-gap configuration moves to a three-gap configuration (via Rule 2) while a three-gap configuration moves to a two-gap configuration (via Rule 3).

Phase 1. Legitimate execution.
$Rule_1 :: (0, 2, z)$ with $z \neq \{0, 1, 2, 3\} \rightarrow (1, 2, z - 1)$
$Rule_2 :: (1, 2, z)$ with $z \neq \{0, 1, 2, 3\} \rightarrow (0, 3, z)$
$Rule_3 :: (0, 3, z)$ with $z \neq \{0, 1, 2, 3\} \rightarrow (0, 2, z + 1)$

3.2 Phase II

Phase II takes care of the execution of the system while the initial configuration is not a legitimate configuration. We identify four different classes of configurations that needs a special attention:

- the symmetric configurations $((x, y, y))$ and
- the asymmetric configurations different from the legitimate one (x, y, z with $x \neq y \neq z$)

Phase II. Execution starting from special configurations.
$Rule_{SC1} :: (0, y, z)$ with $y \neq z \neq \{1, 2, 3\} \rightarrow (0, min(y, z) - 1, max(y, z) + 1)$
$Rule_{SC2} :: (x, y, y)$ with $x \neq y \neq 0 \rightarrow (x, y - 1, y + 1)$
$Rule_{SC3} :: (x, y, z)$ with $x \neq y \neq z, x < y < z \rightarrow (x - 1, y + 1, z)$
$Rule_{SC4} :: (0, 0, z) \rightarrow (0, 1, z - 1)$ when 1 robot executes or $(1, 1, z - 2)$ when two robots execute
$Rule_{SC5} :: (0, 1, z) \rightarrow (0, 2, z - 1)$

3.2.1 Correctness

In the following a round denotes the shortest fragment of execution where each robot executes at least once.

Lemma 1 *Starting in a legitimate configuration, after the execution of a round, the position of all the robots shift one location in the same direction.*

Proof. First we prove that the system started in a legitimate configuration, C_i, always moves to a configuration $C_{i+1 \bmod 3}$ for all $i \in \{0, 1, 2\}$. Without restraining the generality consider the execution starts in a configuration of type C_0 characterized by the tuple $(0, 2, z)$ and let $l_1, l_2, \ldots l_n$ a virtual notation of the n slots of the ring such that l_1 is occupied by the first robot, r_1, l_2 by a second robot r_2 and l_5 by a third robot r_3. In C_0 only the robot r_1 is activatable for the execution of the $Rule_1$. Since the scheduler has to choose at least one robot in each configuration after the execution of r_1 the configuration changes to a configuration of type C_1 characterized by the tuple $(1, 2, z)$. In this configuration only r_2 can execute $Rule_2$ and the system moves to a configuration of type $(0, 3, z)$. In this configuration only r_3 is activatable for the execution of $Rule_3$ and after its execution the system moves to a configuration of type $(0, 2, z)$ with r_1 located at l_n, r_2 located at l_1 and r_3 located at l_4. Note that after the execution of all three robots their position shifted one location to the left. □

In the following we compute the *service time* of the algorithm (the number of steps necessary to all three robots to completely explore the ring at least once).

Lemma 2 *The service time of Algorithm 3.1 is kn.*

Proof. Following Lemma 1, after the execution of a round, robots move one location (they explored exactly one location) and all in the same direction. In order to explore the n locations of the ring, the robots need n rounds. Following the proof of Lemma 1 a single robot can execute in each configuration of a round. Therefore the length of a round equals the number of robots $k = 3$. It follows that the service time of the algorithm is kn. □

Let SC_1 denote the configurations $(0, y, z)$ with $y \neq z \neq \{0, 1, 2, 3\}$, SC_2 denote (x, y, y) and SC_3 denote the configurations (x, y, z).

Lemma 3 *Starting from a configuration SC_i the system converges to a legitimate configuration (C_i, i=1,3).*

Proof. In the following we will examine exhaustively the four classes of configurations.

- Assume the execution starts in a configuration of type $(0, y, z)$. Assume $y < z$ (the other case is symmetric). In this case only one robot may be activated for the execution of the Rule $Rule_{SC1}$ (the one having the distances to the other two robots 0 and y respectively). This robot is the unique robot activatable for the execution of the $Rule_{SC1}$ until y becomes 2. The execution converges in this case to the configuration C_0 in $y - 2$ number of steps.

- Assume the execution starts in a configuration of type (x, y, y) with $x \neq y, y \neq 0$. In this configuration only one robot is activatable for the execution of the $Rule_{SC2}$. The system moves in the configuration $(x, y - 1, y + 1)$ of type (x, y', z) with $y' = y - 1 < z$. If $x = 0$ then the system (via the case 1) converges in $y - 3$ steps to a configuration of type C_0. If

$x \neq 0$ and $x < y-1$ then the system moves to the configuration $(x-1, y, y+1)$ and after the execution of $Rule_{SC3}$ to the configuration $(x-2, y+1, y+1)$ which is a configuration of type $SC2$. It follows that in at most x steps the system converges to a configuration of type C_0. Otherwise, $x \geq y+1$. First consider $x > y+1$, then after the execution of $Rule_{SC3}$ we reach configuration $(y-2, y+1, x)$ and after $y-2$ steps where we apply the $Rule_{SC3}$ we converge to a configuration of type $(0, x, y)$. Finally, consider $x = y+1$, then after applying $Rule_{SC2}$ we reach a configuration $(y-1, y, y+2)$ and then we apply the above reasoning.

- Assume the execution starts in a configuration of type (x, y, z) with $x \neq y \neq z$ and $x < y < z$. In this case the only robot activatable for the execution of the rule $Rule_{SC3}$ is the robot at distances x respectively y from the other two robots. Let r_2 be this robot. After the execution of r_2 the system reaches a configuration of the same type with x reduced by one and y increased by 1. r_2 is activatable in this new configuration until x becomes 0. That is, after x steps the system converges to a configuration $(0, y+x, z)$ of type C_0 if at each increment of y the value stays different from z. In the case when an increment of y moves the system in a configuration of type $CS2$ then after at most x steps the system converges to a configuration of type C_0.

- Assume the execution starts in a configuration of type $(0, 0, z)$ (with $z \geq 7$). In this configuration two robots are activatable for the execution of $Rule_{SP4}$. If the scheduler chooses both robots then the system moves in a configuration of type $CS2$ and then after one step in a configuration of type C_0. If only one of the two robots is allowed to execute, then the system moves in a configuration of type $(0, 1, z-1)$ and after the execution of $Rule_{SP5}$ the system moves to the configuration C_0. Observe that no collision can happen because of asynchrony. This is because the first robot scheduled to move according to the algorithm *Phase I* is the same that may create the collision when we pass from *Phase II* to *Phase I*.

□

4 On the maximum number of robots

Lemma 4 *For any ring of size n it is impossible to solve the perpetual exploration with $n-2$ robots, where $n \geq 2$.*

Proof. In the following we do not use the adversarial power of the scheduler, I.e., we consider that the scheduler activates all the nodes that are expected to move by the algorithm. Consider a ring of size n with $n-2$ robots where $n \geq 2$. For the case $n = 2$ the result is trivial. First, consider the case where $n-2$ is even. Consider the configuration where the robots are grouped into two blocks of size $\frac{n-2}{2}$. Between an extreme node of one block and the closest node of the other block there is an empty node. It is simple to see that no robot can move. This is because the closest nodes of the two blocks have a symmetric view, thus they will both move on the same free node, violating the mutual exclusion constraint.

Now, consider $n-2$ to be odd. Again, we use the two free nodes to separate the nodes into two blocks of different size. Without loss of generality, consider that the algorithm moves the nodes at the extreme of the smallest block, denoted B, to join the extreme of the other block, denoted A. This may happen up to the time we reach a configuration where there is a node isolated and a block of size $n-3$. Then, either this node moves and creates a single block or the robots go back

to rebuild the set B. If the robots are all collected to form a single block, than the extreme of this block will move to occupy the two free nodes, rebuilding two blocks of different sizes. A similar argument can be applied to the case where the algorithm move the nodes from the largest block to the smallest one. It is simple to see that there is at least one robot that does not visit the complete ring. □

Lemma 5 *For any ring of size n, it is impossible to solve the perpetual exploration with $n-k$ robots with $2 < k \leq n$ where $n \bmod k = 0$.*

Proof. Consider a ring of size n and $n-k$ robots with $2 < k \leq n$ where $n \bmod k = 0$. Consider the robots grouped in k blocks of size $\frac{n-k}{k}$ such that each of these block is separated from the successive one in the ring (in both directions) by a free node. It simple to see that we need exactly k free nodes. The configuration described is completely symmetric. Thus, either no nodes move or all the nodes at the extreme of a block move. In this latter case, the node mutual exclusion property is violated. □

Hereafter, we will use the following notation to simplify the presentation: b_z denotes $z \geq 1$ robots located at z consecutive nodes.

Lemma 6 *For any ring of size n, it is impossible to solve the perpetual exploration with $k = n-3$ robots.*

Proof. Consider a ring of size n and a set of $n-3$ robots. Then consider the initial configuration where the $n-3$ robots are arranged on the ring in order to form three blocks respectively two of the same size x, both denoted b_x, and one of size y, denoted b_y (note that for any $n \geq 10$ we can have this configuration). Any pair of blocks is divided by a free node. Now, it is simple to see that the robots belonging to one of the block of size x cannot move towards the other block of size x, because they will collide on the same free node. So consider the case, where the robots on both blocks b_x move to join b_y. It may happen that they are perfectly synchronized, and at each step, exactly two robots (one from each of the above blocks b_x) move. Then, we will eventually join into a single block and come back to the initial configuration. Thus, no robot completely visit the ring.

Finally, consider the case where the robots in b_y move towards the blocks b_x. Since the robots at the extreme of b_y, denoted r_1 and r_2, have a symmetric view, they are both scheduled to move. Now, consider that r_1 is very slow and it still belongs to b_y while r_2 already joined one of the other block. After r_2 joins one b_x, we have three blocks of size x, y and $x+1$. We denote this three blocks b_x, b_y and b_{x+1}

Eventually, r_1 will move because of the old snapshot. Then, we use the asynchrony to decide when r_1 will move and we show that either at the time r_1 moves it generates a collision with another robot; or the system will cycle between few configuration where the same small sets of robots move far and back.

The algorithm should provide the rule to move at least one robot and should avoid that a robot from b_x moves to b_y. Otherwise at that time we move r_1 from b_y to b_x and we create the collision.

Note, that the algorithm can decide either because of the size of the blocks of robots in the current configuration; or according to the fact that one of such blocks is odd or even. In this last case, we choose b_x and b_y odd and even or vice versa, and the claim follows.

So, let consider the size of the blocks. We have the two following cases:

1. $b_x < b_{x+1} < b_y$

2. $b_y < b_x < b_{x+1}$

We show that no matter which is the rule of the algorithm, either the robots go back and forward without visiting the ring or we create a collision.

- max $\rightarrow$ min. Then consider the case (2), we have that a node from b_{x+1} goes back to b_y, At the same time, robot r_1 moves towards b_x which becomes the new b_{x+1}. Then r_1 comes back to b_y and we repeat forever this scenario. Thus no robot visit all the ring.

- min $\rightarrow$ max. Consider case (1) and we have a collision because a node from b_x will move towards b_y colliding with r_1

- max $\rightarrow$ *inter*. We consider the case, (1) let $y = x + k$ with $k \geq 2$. When the first robot moves from b_y to b_{x+1}, r_1 moves to b_x. Then, we have three set of size b_{x+1}, b_{x+2} and b_y. If k is such that b_y becomes smaller than b_{x+1}, then one robot will continually moves from b_{x+2} to b_{x+1} and back. Otherwise, robots will move from b_y to b_{x+2} up to the time this latter become the biggest. Then, the same robots will go back to b_y. Thus, no robot explore the ring completely.

- *inter* $\rightarrow$ max. Consider case (1). A robot moves from b_{x+1} to b_y and at the same time r_1 move to b_x. Then, because we apply the same rule, r_1 comes back to b_y and we repeat the complete scenario infinitely many times. Thus, no robot explore the ring completely.

- *inter* $\rightarrow$ min. We create a collision because of (2).

- min $\rightarrow$ *inter*. Consider the case (1). Then a robot r_2 is scheduled to move from b_x to b_{x+1}. But before r_2 moves towards b_{x+1} r_1 move towards b_x. Thus the two sets are again of the same size, both are b_{x+1}. So, other two robots are scheduled to move from b_y to b_{x+1}. This time, is the robot that move towards the set where r_2 is ready to move that is faster. Then this latter becomes of size $x + 2$ and the other is still of size $x + 1$. Thus, b_{x+1} is the min set and b_{x+2} is the intermediate. (we choose b_y sufficiently big). So a robot r_3 will be scheduled to move from b_{x+1} to b_{x+2}. r_3 will collide with r_2. The claim follows.

It is simple to see, that if at the beginning b_y is equal to b_x or b_{x+1} we can not solve the problem. □

Lemma 7 *For any ring of size n, it is impossible to solve the perpetual exploration with $k = n - 4$ robots.*

Proof. Consider a ring of size n and let the $n - 4$ robots be located in the ring in a such a way that they are divided into two blocks separated by two free nodes on each site. Let b_x and b_y be these two blocks respectively of size x and y. If both blocks are even, it is simple to see that the problem cannot be solved. So, consider an odd number of robots. If b_x is odd then b_y is even or vice versa. The only information that the algorithm may to decide how to move the robots is the parity and the size of the two blocks.

If the algorithm decides to move the robots that are in the smallest group, say b_x, then we consider an initial configuration where the smallest group is even. Otherwise, eventually the block with the biggest size becomes the smallest one, and the robots just come back on their own steps.

So, consider, that the robots move from the smallest block towards the biggest block. It is simple to see that in the worst case, at each step a pair of robots move away from b_x towards b_y. Either any two steps a new pair of robots join b_y or at some point b_x will be split into two blocks of equal size $\frac{x}{2}$, denoted b_{x1} and b_{x2}, such that there are two free nodes between b_{x1} and b_{x2} and both b_{x1} and b_{x2} are separated with one free node from b_y. So either these two new blocks go back to reform b_x or they go towards b_y forming a single block. Once the single block is formed, they will just go back to form again b_{x1} and b_{x2}. Finally, consider the case where the algorithm, once in the configuration with b_{x1}, b_{x2} and b_y let move the robots from b_y to the other blocks. When all the nodes of b_y have being split between b_{x1} and b_{x2}, we have again two blocks one even and the other odd separated by two free nodes in each side. Then we can repeat the above reasoning. Thus, there is at least a robot in b_y that never visits at least one node initially occupied by the nodes in b_x.

Now, consider the case, where the algorithm decides to move the robots belonging to the odd block b_y. The two robots at the extreme of b_y, said r_1 and r_2 take the same snapshot of the network and decide to move towards b_x. But r_2 is much faster than r_1 and execute the first step while this latter is still in b_y. Then r_2 takes a snapshot and sees two blocks and itself. Either it comes back to b_y or it joins b_x. In this latter case, we can build a scenario where r_1 and r_2 go back and forward forever. So consider the case where r_2 to join b_x while r_1 is still in b_y. Now, b_x is odd and b_y is even. So the two robots at the extreme of b_x move towards b_y. These are r_2 and robot r_3 at the other extreme of b_x on the side of r_1. Consider the case, where at the second step of r_3 towards b_y, r_1 moves according to its old snapshot. These two robots will collide on the same node. □

4.1 A Protocol with $n-5$ robots

In this section we propose a protocol that achieves a perpetual exploration of a ring of size $n \geq 10$ with $k = n - 5$ robots where k is odd and $n \bmod k \neq 0$. As for the protocol with 3 robots, the algorithm works into two phases: the first phase is to perpetually explore the ring and the first phase is to reach a legitimate configuration. The algorithm works for any number of robots $k \geq 7$.

The main idea of the algorithm is to simulate the movement of the free nodes, thus reducing to the algorithm presented for 3 robots. Each robot takes a snapshot of the system and according to its view, it decides to move or to remain at its current position. If the robot is selected to move, the algorithm states where the robot has to move. Note that the decision is related to the snapshot taken by the robot and not by the current state of the system. These latter may not coincide.

The snapshot taken by a robot at node u_i (also called view) to one side is the sequence $u_{i+1}, u_{i+2}, \ldots u_{n-1}, u_0, u_1 \ldots u_{i-1} u_i$ (similarly on the other side). So we consider the robot itself to be counted as the robot in the last node of the sequence.

Each time a robot moves, it moves into one of the adjacent node that is free according to its snapshot. The algorithm is described in terms of rules. As an example consider the view $C^i = (0, 0, 0, b_2, 0, 0, b_y)$, the robot that has this view is the last one (on our right) in the block b_y. So this will be the one to move, and since it has an adjacent free node only on one side, it will move to that free node and get closer to b_2.

We specify in which direction the robot should move to disambiguate.

The main idea of the algorithm, once in a legitimate configuration is to let move a node from the biggest set to the smallest one (via the longest path) up to the time the size of the smallest becomes 3 ($Rule_{LC1}$, $Rule_{LC2}$ and $Rule_{LC3}$). Then the robot in the block b_3 in the opposite side of the robot just arrived, move towards b_y via the shortest path, $Rule_{LC4}$ and $Rule_{LC5}$.

	View	Action
$Rule_{LC1}$	$C^i = (0, 0, 0, b_2, 0, 0, b_y)$	$\rightarrow$ move towards b_2 (via the longest path)
$Rule_{LC2}$	$C^i = (0, 0, b_2, 0, 0, b_y, 0, b_1)$	$\rightarrow$ move towards b_2
$Rule_{LC3}$	$C^i = (0, b_2, 0, 0, b_y, 0, 0, b_1)$	$\rightarrow$ move towards b_2
$Rule_{LC4}$	$C^i = (0, 0, b_y, 0, 0, 0, b_3)$	$\rightarrow$ move towards b_y (via the shortest path)
$Rule_{LC5}$	$C^i = (0, b_y, 0, 0, 0, b_2, 0, b_1)$	$\rightarrow$ move towards b_y

Table 1: Phase 1. Algorithm at node u_i where the number of robots are at least 7.

Phase 2. We now describe how to reach a legitimate configuration starting from any configuration.

Rule 1. If the configuration has just two blocks, denoted b_x and b_y, then b_x and b_y are separated by two paths of free nodes of different size (i.e., either of 2 and 3 free nodes or of 1 and 4 free nodes). Since we consider a number of robots which is odd, the two blocks are of different size. Without loss of generality, let b_x be smaller than b_y. Now if the size of $1 < x < 4$ we have to ensure that the two blocks are at the right distance.

If the size of b_x is bigger than 3 than the robot at one extreme of b_x takes the shortest path towards b_y. Note that if the two blocks are both composed of more than one robot, at each step we can identify exactly one robot to move, that is the one that goes from one block to the other. This is possible because both the two blocks and the two paths of free nodes have different sizes. If $x = 1$ then one robot moves from b_y towards b_x taking the longest path, i.e. the path with at least three free nodes. Then, in one step we are in a configuration, which is a special case of three blocks (see later). In that case, we use Rules 3. whose application eventually lead to the creation of two blocks of size greater than 1.

Once b_x and b_y have the right size, if these latter are divided by 1 free node from one side and 4 free nodes from the other, robots in b_y move one by one, shifting of a position towards b_x in the path of 4 free nodes (see Rule 3.$(a)ii$). The following rules define the algorithm to let the robots to arrange into two blocks without colliding, no matter in which configuration they start to execute.

Rule 2. the robots are divided in more than 3 blocks. These means that it exists at least a pair of blocks at distance one from each other (they are separated by a free node). Then, when two blocks are at distance one, the robots belonging to the smallest block shift one after the other to join the biggest block.

Rule 3. If we reach a configuration with 3 blocks, respectively b_x, b_y and b_z apply the following rules:

(a) If b_x and b_y are divided by three free nodes then, b_z is at distance one from both of them, i.e., $(b_x, 0, 0, 0, b_y, 0, b_z, 0)$.

i. $z = 1$, the single robot in b_z moves to b_y with $y \geq x$ (if $x = 2$ we are in the algorithm *Phase I*).

ii. $z > 1$, one robot from b_z moves to b_x with $1 \leq x \leq y$ (i.e., if $x = y$ two robots move from b_z each of them to join one of the other blocks).

(b) Otherwise, we have the following configuration $(b_x, 0, b_y, 0, 0, b_z, 0, 0)$.

For $z > 1$:

i. if $x = y = 1$, both single robots respectively in b_x and in b_y move one step towards b_z;

ii. if $1 < x < y$, one robot moves from b_x to b_y;

iii. if $1 = x < y$ executes the algorithm *Phase I* if either y or z equal 2. Otherwise apply Rule 1.

For $z = 1$:

i. $1 < x < y$ move the single robot in b_z to b_x (special case of Rule 1.).

ii. $1 = x < y$ the robot in b_z moves a step towards b_y.

Rule 4. the robots are all collected to form a single block, $(0, 0, 0, 0, 0, b_z)$. The robots at the extreme are scheduled to move one step far. Either, both robots move and then we successively apply Rule 3. (a) or one of them move and we apply Rule 1. Let r_1 be the robot that has not moved yet. Note that the robot that is scheduled to move because of Rule 1. is r_1. Analogously for Rule 3.(a).

4.2 Correctness

A legitimate configuration is one of the configuration we reach in the execution of algorithm *Phase I*.

Lemma 8 *Starting from a non legitimate configuration the system converges to a legitimate configuration in finite time.*

Proof. It is simple to see that once robots are divided into two blocks composed of more than 1 robot, we can safely manage these two blocks to adjust their sizes. It remains to show that we are able to divide the robots into these two blocks, no matter which is the initial configuration. Once we have a single block eventually, the block is divided into two groups, because of Rule 4, Rule 1. and Rule 3.$(a)ii$. When we have more than 3 blocks because of Rule 2. and Rule 3. we let the blocks to join each other. Since we maintain the invariant that robots move from the smallest to the biggest block, no collision may happen in this phase.

Finally, consider the configuration with 3 blocks. It simple to see that when there is a single robot and the other two groups are greater, then we are either in a legitimate configuration or we apply Rule 1. and Rule 3.$(a)i$ to adjust the size of the blocks and we eventually converge on a legitimate configuration.

The only special case, is when we have two single robots and a third block. If the two single robots are at distance of three free nodes, then we apply Rule 3.$(a)ii$ and eventually we obtain two blocks whose size is greater than 1.

If the two robots are at distance one from each other, according to Rule 3.$(b)i$ they move one step towards the third block. Then, we reach a configuration where the two single robots are at distance of three free nodes, and we apply Rule 3.$(a)ii$.

Finally, if the two robots are at distance 2 from each other, we apply Rule 3.$(b)ii$ for $z = 1$ and we reach again the configuration where we can apply Rule 3.$(a)ii$. This configuration can also be reached started from the configuration immediately above because, due to the asynchrony, one of

the robots move while the other is still in the previous position. Note that there is no way for the robots to collide, because when they move they remain at distance one from the block. Analogously for the robots that move from the block to the single robots.

It is simple to see that at most after $2k$ steps, where k is the number of robots, the robots are divided into two blocks or are in a legitimate configuration. This is because at most after 2 steps a robot moves from one block to the other. Once we have the two groups, in at most other $2k$ steps, we obtain a legitimate configuration. □

It remains to prove that once the robots reach a legitimate configuration, no collisions will happen at some point during the exploration because of the asynchrony.

Lemma 9 *The Algorithm implements the exclusive perpetual exploration.*

Proof. Because of Lemma 8, the system in a finite number of steps converges to a legitimate configuration.

We do not risk a collision even though a node with an old snapshot will move when the other robots are already executing the *Phase I* of the algorithm. This is because the same robot will be also the one scheduled to move according to the algorithm *Phase I*. Then either this node finally joins the adjacent block and we obtain two blocks, or the algorithm *Phase I* starts from this configuration. Note that once we have two blocks of size greater than one, a robot at a time move. Then we will have no more problem of asynchrony and we will reach a quiescent legitimate configuration, i.e., a legitimate configuration where no robot is expected to move because of a previous snapshot.

Then, consider that at time t the system is in a quiescent legitimate configuration c, i.e., there is no robot that at that time should move because of a Look phase executed before t. It is simple to see that once we reach a quiescent legitimate configuration the system will move to another quiescent legitimate configuration. This is because in any legitimate configuration, each robot has a different view of the system and thus, at a given time, only one robot moves.

If $x = 2$, even though one robot moves from b_y to b_x, we maintain the invariant that $y' = y - 1 > x$. When $x = 3$ we move one robot from b_x to b_y. Then we return to the initial configuration. Note that once a robot moves from b_y to b_x via the longest path, it will eventually come back to b_y through the shortest path and shift in b_y up to reach the initial position and repeat all the above steps. Then, each robot visits all the nodes in the ring. Hence, the claim follows. □

5 Concluding remarks

We investigated the problen of exclusive perpetual exploration of a ring by a team of mobile robots that do not have sense of direction. We presented tight results both for the case of a team of minimal cardinality and of maximal cardinality. The exclusion contraint makes the maximum cardinality problem harder to solve. Our work raises several important open questions:

1. Would it be possible to generalize our approach to other regular topologies (*e.g.* torus) where no sense of direction is available ?

2. Would probabilistic coin tossing help in cases where we proved deterministic solutions are impossible ?

References

[1] Roberto Baldoni, François Bonnet, Alessia Milani, and Michel Raynal. On the solvability of anonymous partial grids exploration by mobile robots. In Theodore P. Baker, Alain Bui, and Sébastien Tixeuil, editors, *OPODIS*, volume 5401 of *Lecture Notes in Computer Science*, pages 428–445. Springer, 2008.

[2] Stéphane Devismes, Franck Petit, and Sébastien Tixeuil. Optimal probabilistic ring exploration by asynchronous oblivious robots. In *Proceedings of Sirocco 2009*, Lecture Notes in Computer Science, Piran, Slovenia, May 2009. Springer-Verlag Berlin Heidelberg.

[3] Paola Flocchini, David Ilcinkas, Andrzej Pelc, and Nicola Santoro. Computing without communicating: Ring exploration by asynchronous oblivious robots. In *OPODIS*, pages 105–118, 2007.

[4] Paola Flocchini, David Ilcinkas, Andrzej Pelc, and Nicola Santoro. Remembering without memory: Tree exploration by asynchronous oblivious robots. In Alexander A. Shvartsman and Pascal Felber, editors, *SIROCCO*, volume 5058 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2008.

[5] Ralf Klasing, Adrian Kosowski, and Alfredo Navarra. Taking advantage of symmetries: Gathering of asynchronous oblivious robots on a ring. In *OPODIS*, pages 446–462, 2008.

[6] Ralf Klasing, Euripides Markou, and Andrzej Pelc. Gathering asynchronous oblivious mobile robots in a ring. *Theor. Comput. Sci.*, 390(1):27–39, 2008.

[7] Giuseppe Prencipe. Instantaneous actions vs. full asynchronicity : Controlling and coordinating a set of autonomous mobile robots. In Antonio Restivo, Simona Ronchi Della Rocca, and Luca Roversi, editors, *ICTCS*, volume 2202 of *Lecture Notes in Computer Science*, pages 154–171. Springer, 2001.

[8] Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM J. Comput.*, 28(4):1347–1363, 1999.

A Impossibilities on chains

Observation 1 *There is no terminal configuration,* i.e.*, in every configuration, at last one robot must move.*

Lemma 10 *Exclusive perpetual exploration on a chain with one robot is impossible.*

Proof. Consider a chain of size $n \geq 3$ with one robot. By Observation 1, the robot must move when the scheduler activates it.

Let $v_1, v_2, \ldots, v_n$ denote the sequence of nodes that constitute the chain. Upon activation, the robot is located at node v_j with $j \in \{1, \ldots, n\}$ and its view is such that it has $j-1$ nodes on one direction and $n-j$ on the other one. In other words, the position of the robot on the chain defines two sub-chains, respectively $v_0, \ldots, v_{i-1}$ and $v_{i+1}, \ldots, v_n$. The algorithm can decide to move the robot either towards the direction with the longest sub-chain or to the one with the shortest sub-chain. Then, the robot moves accordingly.

First, we consider the case where the algorithm does not change the way it takes its choices (called policy), i.e., if at time t the algorithm decides that the robot has to move towards the longest sub-chain, there is not a time $t^{'} > t$ where the scheduler decides that the robot has to move towards the shortest sub-chain.

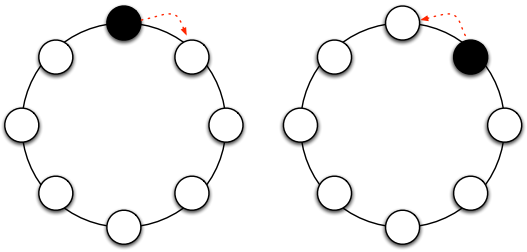
Consider the case where the robot moves to the direction with the longest sub-chain. Then, after the robot visits the node $\lceil \frac{n}{2} \rceil$, the direction with the longest sub-chain is the one where the robot has just come from. Thus, the scheduler moves the robot back and the remaining nodes are not visited. Hence, the result.

Consider the case where the robot moves to the direction with the shortest sub-chain. This may not happen if the robot is at node v_0 and v_1, otherwise the robot will be simply stuck at that node. Thus, without loss of generality, consider the robot to be located at v_0. The algorithm can only move the robot to node v_1. Once at v_1 the shortest sub-chain is composed by node v_0. If the algorithm decide to let the robot move to the direction with the shortest sub-chain, it is simple to see that the robot will continuously move from v_0 to v_1 and back. Hence, the result.

To complete the proof, consider the algorithm to change its policy, once the robot is at some node v_i with $i \in 1, \ldots, n$. Once at v_i, since the robot is oblivious, the robot does not know if it arrived at v_i from node v_{i-1} or from node v_{i+1}. In both cases it will take the same decision. So, if up to v_i the robot moved to the direction with the shortest sub-chain, then changing the policy leads the robot to come back on its steps. Otherwise, up to v_i the robot moved to the direction with the longest sub-chain. But then, once the algorithm changes its policy at node v_i, if $v_0, \ldots v_{i-1}$ is the shortest sub-chain, the robot comes back to its steps. Otherwise, consider the case where the robot arrived at v_i from node v_{i+1}. Because of the indistinguishability, the robot will come back through the nodes it previously visited. In both cases, the robot will not visit the complete chain. Hence, the claim holds. □

Lemma 11 *Exclusive perpetual exploration on a chain is impossible.*

Proof. It is trivial to see that the perpetual exploration on a chain with more than one robot is impossible. This is because either at some point two robots traverse the same edge at the same time, thus violating the mutual exclusion property; or each robot does not visit some part of the chain. Hence, the result follows from Lemma 10. □



(a) Moving Left (b) Moving Left

Figure 1: Impossibility with one robot

B On the minimal number of robots

Our first results related to the possibility of exclusive perpetual exploration with a minimal number of robots show that at least three robots are necessary.

Lemma 12 *Perpetual exploration on a ring of size $n \geq 3$ with one robot is impossible.*

Proof. Consider a ring of size $n \geq 3$ with one robot. By Oservation 1, the robot must move when the scheduler activates it. Upon activation, the view of the robot is symetric, so the scheduler can decide about left and right and make the robot move to the right (Figure 1(a)). The resulting configuration is isomorphic to th previous one. The scheduler now chooses that the robot goes left (Figure 1(b)). The scheduler ten repeats the process. As a result, there exists an admissible execution suc that the robot explores only two nodes. As $n \geq 3$, at least one node remains unexplored, hence the result. □

Lemma 13 (Flocchini et all. [3]) *Algorithm of Ring Exploration allows a team of k robots to explore a n-node ring and enter a terminal state with in?nite time, provide $dgcd(n, k) = 1$.*

Let us now turn our attention to the case of three robots in a ring. We will show that perpetual exploration by three robots is impossible in a ring of less than ten nodes. To prove this claim, let us define the notion of *configuration*.

Definition 1 *For three robots in an n-node ring, a* configuration *is a triple of positive integers (x, y, z) corresponding to the respective distances between two consecutive robots in the ring. (Hence $x + y + z = n - 3$).*

A consequence of this definition is that if a configuration $\mathcal{C}$ is symmetric, i.e., if $\mathcal{C} = (x, x, x)$, then, in view of lemma 13, perpetual exploration is impossible. A configuration of the form (x, y, y) with $x \neq y$ is called *semi-symmetric*, and a configuration of the form (x, y, z), with x, y, and z pairwise distinct, is called *asymmetric*. The *view* of a robot is a non-ordered pair of integers specifying the distance between the robot and the two other robots in the ring. For instance, in a semi-symetric configuration (x, y, y), one of the robots has view $\{y, y\}$ while the two other robots have the same view $\{x, y\}$. A view $\{y, y\}$ is called *symmetric*.

Lemma 14 *To achieve perpetual exploration, any exclusive exploration protocol $\mathcal{P}$ satisfies that, for every robot R, there exists a semi-symmetric configuration in which R has a symmetric view, and $\mathcal{P}$ moves R.*

Proof. Assume, for the purpose of contradiction that, for all semi-symmetric configurations in which R has a symmetric view, the exploration protocol $\mathcal{P}$ does not move R. Let $\mathcal{C} = (x, y, y)$ be semi-symmetric configuration in which two robots, say S and T, have the same view $\{x, y\}$, while robot R has the symmetric view $\{y, y\}$. In such a configuration, since $\mathcal{P}$ does not move R, $\mathcal{P}$ must move S or T. Since S and T have the same view, if $\mathcal{P}$ moves one of the two robots, then it also moves the other robot. Now, the scheduler decides which robot to move. In particular, it can decide to move the two robots S and T simultaneously. In that case, after one round, these two robots will have the same view again, which is either $\{x + 2, y - 1\}$, or $\{x - 2, y + 1\}$, but the same for both robots. The two corresponding configurations are semi-symmetric, in which R has again a symmetric view. As a consequence, if R does not move in semi-symmetric configurations in which it has symmetric view, then R always stays at the same node, and exploration cannot be achieved. □

Lemma has a direct consequence for the case of two robots since if n is coprime with 2, no robot can have a symmetric view and thus no exclusive exploration protocol can exist.

Corolary 1 *Perpetual exploration with two robots is impossible.*

Theorem 1 *In the n-node ring, perpetual exploration with three robots is impossible whenever* $n < 10$.

Proof. The proof proceeds by considering several cases, corresponding to different configurations, and different number of nodes. We consider the four cases $n = 4, 5, 7, 8$. The cases $n = 3, 6, 9$ do not deserve to be considered since these number of nodes are multiple of the number of robots, in which case perpetual exploration is impossible (cf., Lemma 13.)

If $n = 4$, then there is only one configuration: $(0, 0, 1)$. In this configuration, assume that the robots R_1 and R_3 have the same view. Hence, if the exploration protocol moves one of them, then it moves both of them. Since there is only one unoccupied node just before R_1 and R_3 move, their simultaneous moves will bring them to this same empty node, causing collision (since the model specifies that at most one robot can occupy a same node simultaneously). Therefore perpetual exploration by three robots is impossible in the 4-node ring.

If $n = 5$ then the are two semi-symmetric configurations: $(0, 0, 2)$ and $(1, 1, 0)$. Assume that the initial configuration is $(0, 0, 2)$, and that the robots with the same view are R_1 and R_3 (see Figure B-A). R_2 has the symmetric view $\{0, 0\}$. Hence, the protocol does not move R_2 since otherwise the scheduler can decide to let only this robot move, which will cause collision. Thus only R_1 and R_3 move. The scheduler can decide to move these robots simultaneously, yielding the following configuration $(1, 1, 0)$ (see Figure B-B) in which robots R_1 and R_3 still have the same view. Now robot R_2 can move. Actually, in view of Lemma 14, robot R_2 must move. When R_2 moves, the scheduler can decide to move it toward R_1. This yields the semi-symmetric configuration $(0, 0, 2)$ again, in which R_2 and R_3 have the same view (see figure B-C). In this configuration, R_1 cannot move, and thus R_2 and R_3 have to move (see figure B-D). Again, the scheduler can move both of them simultaneously, yielding the semi-symmetric configuration $(1, 1, 0)$ in which, in view of Lemma 14, R_1 must move. The scheduler can then moves R_1 toward R_2. Hence, the three robots are back at the same initial positions, without having explored all nodes. Hence perpetual exploration by three robots in the ring of 5 nodes is impossible.

The cases $n = 7$ and $n = 8$ are treated similarly. They both lead to tedious case-by-case analyses, which are skipped in this extended abstract. □

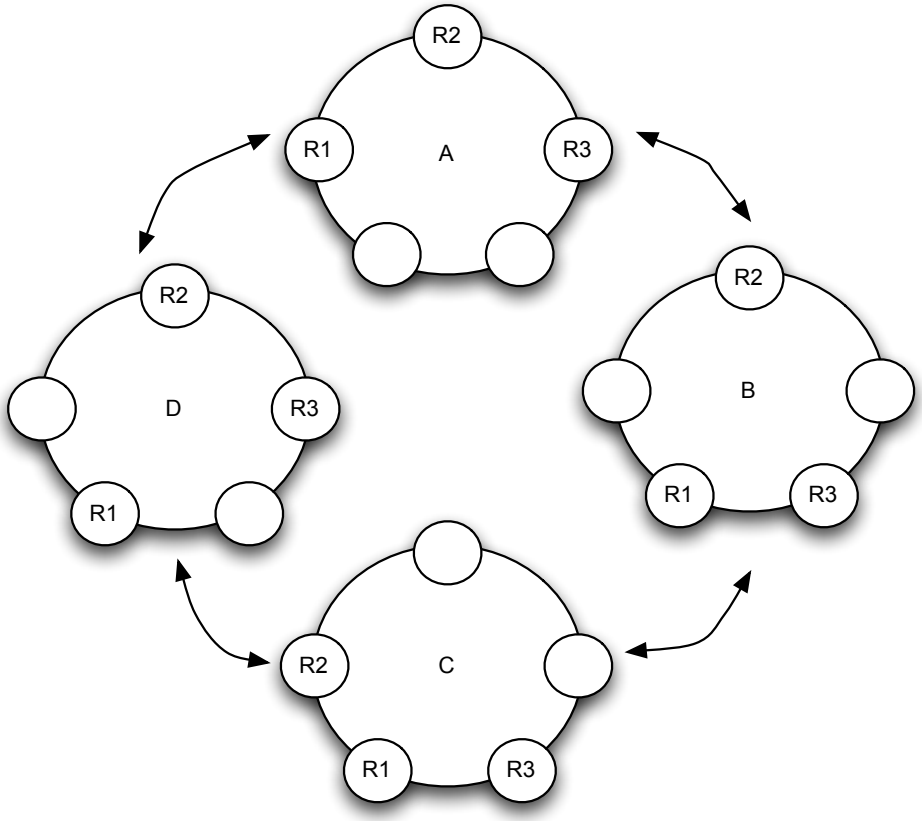


Figure 2: Impossibility